Remarks

A. Claims in the Case

Claims 1-37 and 40-51 are pending. Claims 1, 15, 33, 40, 41, and 47 have been amended. Claims 38 and 39 have been cancelled.

B. The Claims Are Not Obvious Over Copeland in View of Underwood and Kelly Under 35 U.S.C. §103(a)

Claims 1-32 and 47-51 were rejected under 35 U.S.C. §103(a) as being obvious over U.S. Patent No. 5,946,694 to Copeland et al. (hereinafter referred to as "Copeland") in view of U.S. Patent No. 5,873,066 to Underwood et al. (hereinafter referred to as "Underwood") and further in view of U.S. Patent No. 5,806,042 to Kelly et al. (hereinafter referred to as "Kelly"). Applicant respectfully disagrees with these rejections.

In order to reject a claim as obvious, the Examiner has the burden of establishing a *prima facie* case of obviousness. *In re Warner et al.,* 379 F.2d 1011, 154 USPQ 173, 177-178 (CCPA 1967). To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974), MPEP § 2143.03.

Amended claims 1, 15, and 47 describes a combination of features including, but not limited to: "identifying an inheritable class of objects to represent the one or more conditions of a reinsurance contract, wherein the reinsurance contract is represented by an reinsurance contract object, wherein the reinsurance contract object is a parent of a section object; creating an instance of the inheritable class of objects to identify a condition object, wherein the condition object is a child of the section object". The cited references do not appear to teach or suggest the above-quoted feature of claims 1, 15, and 47, in combination with the other features of the claims.

The Office Action relies on Copeland for the above-quoted features of claims 1, 15, and

47. The Office Action states: "Examiner is relying on the 'mixin object' for the recited 'section object'". The Office Action further states: "Examiner is relying on the 'data object' for the recited 'condition object'".

Copeland states:

> The system and infrastructure related functions necessary for the business objects to interact with the computer system are embodied in classes of objects collectively termed mixin objects. The mixin object contains a model of the various system functions and descriptions of how business objects interact with them. One specific example of a mixin object is a mixin object created specifically for a relational database. This relational database mixin object would provide the interface necessary to interact with the computer system and infrastructure related concerns.
>
> For example, the mixin object provides the necessary interaction with the system-level security service to prevent the unauthorized invocation of a business method if the user was either not authenticated nor authorized to invoke a particular method. The relational database mixin object might also interact with the database to manage the connections to the database (open, close, suspend, resume, etc.) and also to manage the flow of data between the database and the object that requested the data from the database. This includes retrieving data from the database at the beginning of a transaction and storing the data at the end of the transaction. The mixin object may also control the memory paging of the object and can determine when the object is readily available in memory and when it is not. This allows system-level resources like memory, communication ports, processor cycles, database connections, etc. held by that object to be released when the object is not expected to perform any actions for some time. (Copeland, column 6, lines 33-60)

Copeland appears to teach "mixin objects" that embody system and infrastructure related functions. The mixin objects may, for example, provide interaction with the system-level security service, manage the connections to a database, manage a flow of data between a database and a requesting object, and control memory paging of an object.

Copeland also states:

> Data Object. The data object is another component of the MOA. The data object contains some knowledge of the business domain aspects of the MOA and some knowledge of the infrastructure domain of the underlying system. The data object

> performs a mapping function between the abstract logic of the business domain and the physical data storage components of the system. The data object encapsulates information about the persistence characteristics of the MOA and may also store data values. One example of a data object is an object that contains Structured Query Language (SQL) statements for retrieving specific data from a database. Each data object implements an abstract interface for the business object attributes. Each implementation is specific to a given database product or legacy system (examples include dB2, CICS, IMS, and MQ).
> (Copeland, column 5, lines 20-35)
>
> The actual storing and retrieving of the data from the data base is provided in classes of objects collectively termed data objects. The data objects contain the logic needed to map between any data required to perform the business logic and the physical data which is stored in a secondary storage location or back end store, like a database, a file system or an application system. One example of a data object is an insurance policy data object that provides the necessary data for an insurance policy business object where the necessary data is stored in a relational database. This insurance policy data object may provide an interface or access to the relational database for create, retrieve, update, and delete (CRUD) operations. However, the interface and methods used by the business object to perform the various operations, such as setting and retrieving attributes, do not access the relational database directly, but rather caches the information for use in the CRUD methods which are performed by the data object.
> (Copeland, column 6, line 61 to column 7, line 11)

Copeland also appears to teach "data objects" that provide for storage and retrieval of data from a database. The data objects may contain logic to map between data required to perform business logic and physical data in a secondary storage location.

With respect to condition objects, which the Office Action equates with data objects, Applicant's specification states:

> In one embodiment, the reinsurance transaction processing software may include an object-oriented framework, the condition component framework. This framework, i.e., the condition component framework, may permit the addition and/or modification of condition components of the reinsurance contract. These components may comprise, for example, premium limits, consolidation conditions, and other conditions for reinsurance contracts. In one embodiment, the components may be implemented as reusable objects.
> (Specification, page 41, line 26 - page 42, line 2)

The "data objects" of Copeland do not appear to be comparable to the "condition objects" of applicant's claims. The data objects of Copeland appear to include data that allows a program to map data to the appropriate physical storage unit. Applicant; submits that the condition components of Applicant's claims objects that include data related to the condition of a contract. Applicant submits that the data objects of Copeland are not the same as Applicant's condition objects.

Additionally, Copeland does not appear to teach or suggest that the data objects are children of the mixin objects. Instead Copeland appears to teach that mixin objects and data objects exist as siblings with respect to each other, not in a parent child relationship. For example, Copeland states that "A Managed Object Assembly (MOA) is an object instance that is comprised of an instance of a data object, a mixin object, and a managed object". (Copeland, col. 5, lines 51-54).

Applicant submits that Copeland does not teach or suggest identifying an inheritable class of objects to represent the one or more conditions of a reinsurance contract, wherein the reinsurance contract is represented by an reinsurance contract object, wherein the reinsurance contract object is a parent of a section object, and creating an instance of the inheritable class of objects to identify a condition object, wherein the condition object is a child of the section object, in combination with the other features of claims 1, 15, and 47.

Applicant submits that, for at least the reasons discussed above, claims 1, 15, and 47 and the claims depending thereon are patentable over the cited art. Applicant therefore respectfully requests removal of the 35 U.S.C. §103(a) rejections of these claims.

Applicant submits that many of claims dependent on claims 1, 15, and 47 are independently patentable. For example, claim 9 recites: "wherein configuring the properties and the methods of the condition object consistent with the reinsurance contract comprises: identifying a new condition of the inheritable object class, wherein the one or more conditions excludes the new condition; identifying a new subclass of objects to the reinsurance contract

class of objects; creating a new component object by instantiating the new subclass of objects, wherein the new component object is a child object to the reinsurance contract object." The cited art does not appear to teach or suggest at least these features of claim 9, in combination with the other features of the claim.

The portions of Copeland cited in the Office Action for the above-quoted feature state:

> Managed object 216 contains logic related to the business logic-related functional aspects of MOA 123. For example, a managed object 216 for an insurance policy system may have a method that computes the expiration date of the insurance policy. In addition, managed object 216 contains the logic necessary to route method calls made on MOA 123 to mixin object 212 and the business logic within managed object 216.
> (Copeland, column 9, lines 60-67)

> One example of a managed object for the insurance policy scenario as described above is a managed object that uses data stored in a relational database, where the business object and the data object are both implemented in C++. In this example, the managed object delegates business methods to the business object and, infrastructure methods to the mixin object. Any C++ language specific requirements are handled by the managed object itself. In this example, the managed object uses three predetermined patterns for implementation of its methods. Each of these three patterns is described below.
>
> The pattern used for implementing the business methods is for the managed object to call methods on the mixin object before and after calling the business method on the business object. This procedure allows the mixin object to perform the system-related functions necessary for every method request. It is important to note that this is necessary because the business methods do not include any system level or infrastructure related activities. These system related activities include things like performing security checks, locking records, etc. that need to be performed before the business object performs its method.
> (Copeland, column 7, lines 28-49)

Copeland appears to teach a managed object that delegates business methods to a business object and infrastructure methods to a "mixin object". The managed object calls on the mixin object before and after calling the business method on the business object. Copeland does not appear to teach or suggest identifying a new condition of an inheritable object class, wherein one or more conditions excludes the new condition; identifying a new subclass of objects to the reinsurance

contract class of objects; creating a new component object by instantiating the new subclass of objects, wherein the new component object is a child object to a reinsurance contract object.

Claim 10 recites: "wherein the protection class comprises a proportional protection assignment subclass or a non-proportional protection assignment subclass." The cited art does not appear to teach or suggest at least this feature of claim 10, in combination with the other features of the claim.

The portions of Copeland cited in the Office Action for the above-quoted feature state:

> For example, the mixin object provides the necessary interaction with the system-level security service to prevent the unauthorized invocation of a business method if the user was either not authenticated nor authorized to invoke a particular method.
> (Copeland, column 6, lines 43-47)

Copeland appears to teach a "mixin object" that prevents unauthorized invocation of a business method if a user is not authenticated or not authorized to invoke a particular method. Copeland does not appear to teach or suggest a protection class comprising a proportional protection assignment subclass or a non-proportional protection assignment subclass.

Claim 11 recites: "wherein the section classification class comprises properties, wherein the properties describe a country, a main class of business and a class of business associated with the section classification class." The cited art does not appear to teach or suggest at least these features of claim 11, in combination with the other features of the claim.

The portions of Copeland cited in the Office Action for the above-quoted feature state:

> The mixin object will typically provide support for various system level services such as concurrency, persistence, transactions, etc. The mixin object is created with the ability to understand and work with the infrastructure domain of the system but has no knowledge of the operational functionality of the software application as represented by the business domain logic. It is important to note that the mixin object provides an interface to the infrastructure methods that will be used to store data and other infrastructure domain-related activities.
> (Copeland, column 5, lines 44-51)

Copeland appears to teach a "mixin object" that provides support for system level services such as concurrency, persistence, and transactions. Copeland does not appear to teach or suggest a section classification class comprising properties describing a country, a main class of business and a class of business associated with the section classification class.

Claim 27 describes a combination of features including, but not limited to: "one or more life cycle phase objects derived from the multi-dimensional reinsurance contract framework, wherein each life cycle phase object is a child of one of the insured period objects". The cited references do not appear to teach or suggest the above-quoted feature of claim 27, in combination with the other features of the claim. Examples of life cycle phase objects as recited in claim 27 may be found on page 40, lines 18-25 of Applicant's specification, which state:

> In one embodiment, a life cycle phase object 552 may be created by instantiating a life cycle phase class. In one embodiment, a life cycle phase class may be a concrete subclass of a hierarchy of life cycle phase classes. Typical life cycle phases included in a reinsurance contract may include new agreement offered, offer accepted, offer declined, new quote requested, renewal offered, etc. In one embodiment, the life cycle phase object 552 may own all terms and conditions, all history and all child objects for one phase of one calendar period of the reinsurance contract.

Copeland states:

> The introduction of a managed object assembly and treating it as a single entity affects the traditional life cycle model of objects. In the insurance policy example above, the C++ life cycle model used to create and eliminate the object is insufficient for creating and destroying the managed object assembly. This life cycle deficiency is remedied by using a factory to create the managed object assemblies and by implementing a new method (the "removes()" method) on the object.
> (Copeland, column 8, lines 5-12)

Copeland appears to teach using a factory to create managed object assemblies to remedy purported deficiency in the C++ life cycle model. The "life cycle model" appears to relate to a life cycle software development model, not to a life cycle phase of a reinsurance contract. Copeland does not appear to teach or suggest life cycle phase objects derived from a multi-dimensional reinsurance contract framework, wherein each life cycle phase object is a child an

insurance period object, in combination with the other features of claim 27. Applicant therefore respectfully requests removal of the 35 U.S.C. §103(a) rejection of claim 27, and the claims dependent thereon.

C. The Claims Are Not Obvious Over Underwood in View of Copeland and Kelly Under 35 U.S.C. §103(a)

Claims 33-46 were rejected under 35 U.S.C. §103(a) as being obvious over Underwood in view of Copeland and further in view of Kelly. Applicant respectfully disagrees with these rejections.

Claim 33 has been amended to include features of cancelled claims 38 and 39. Amended claim 33 describes a combination of features including, but not limited to: "wherein each insured period object comprises one or more life cycle phase objects, and wherein each life cycle phase object identifies a particular phase in a life cycle of the particular reinsurance contract during the particular time period." The cited references do not appear to teach or suggest the above-quoted features of claim 33, in combination with the other features of the claim.

Regarding claim 39, the Office Action states:

> Underwood does not explicitly teach that each insured period object comprises one or more life cycle phase objects, wherein each life cycle phase object identifies a particular phase in a life cycle of the particular reinsurance contract during the particular time period. Copeland teaches such a life cycle phase object feature (see column 8, lines 5-21)

Applicant respectfully disagrees that Copeland teaches a life cycle phase object feature. Copeland states:

> The introduction of a managed object assembly and treating it as a single entity affects the traditional life cycle model of objects. In the insurance policy example above, the C++ life cycle model used to create and eliminate the object is insufficient for creating and destroying the managed object assembly. This life cycle deficiency is remedied by using a factory to create the managed object assemblies and by implementing a new method (the "removes()" method) on the object. Factories are used by clients to create objects while the removes() method is used to destroy them. Inside the factory, the components of the managed object

> assembly are created and joined. However, when the managed object assembly is returned to the client, only a single entity is returned. When the client wishes to destroy the object it uses the removes method. The remove() method is an infrastructure method that is delegated to the mixin object and the mixin object takes responsibility for destroying the various components of the managed object assembly.
> (Copeland, column 8, lines 5-21)

Copeland appears to teach using a factory to create managed object assemblies to remedy purported deficiency in the C++ life cycle model. The "life cycle model" appears to relate to a software development model, not to a life cycle phase of a reinsurance contract. Copeland does not appear to teach or suggest an insured period object comprising one or more life cycle phase objects, and wherein each life cycle phase object identifies a particular phase in a life cycle of a reinsurance contract during the particular time period, in combination with the other features of claim 33.

Applicant submits that, for at least the reasons discussed above, amended claim 33 and the claims depending thereon are patentable over the cited art. Applicant therefore respectfully requests removal of the 35 U.S.C. §103(a) rejections of these claims.

Applicant submits that many of claims dependent on claim 33 are independently patentable. For example, amended claim 40 recites: "wherein each life cycle phase object comprises one or more section objects, wherein the one or more section objects are arranged in a hierarchy starting with a main section, wherein each section object comprises children section objects." The cited art does not appear to teach or suggest at least these features of claim 40, in combination with the other features of the claim.

Amended claim 41 recites: "wherein each life cycle phase object comprises one or more amendment objects, wherein the one or more amendment objects are operable to amend one or more condition objects, wherein the one or more amendment objects are shared amongst the one or more life cycle phase objects within the particular time period." The cited art does not appear

to teach or suggest at least these features of claim 41, in combination with the other features of the claim.

D. **Summary**

Based on the above, Applicant submits that the claims are now in condition for allowance. Favorable reconsideration is respectfully solicited.

If any extension of time is required, Applicant hereby requests the appropriate extension of time. Should any fees be required, or if any fees have been overpaid, the Commissioner is authorized to appropriately charge or credit those fees to Meyertons, Hood, Kivlin, Kowert & Goetzel Deposit Account No. 50-1505/5053-28501/EBM.

Respectfully submitted,

Eric B. Meyertons
Reg. No. 34,876

Attorney for Applicant(s)

MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8800 (voice)
(512) 853-8801 (facsimile)

Date: March 21, 2005